DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

# Sistemi Digitali

## VHDL: Esercizi su semplici strutture logiche

Marta Bagatin, marta.bagatin@unipd.it

Corso di Laurea in Ingegneria dell'Informazione

Anno accademico 2022-2023

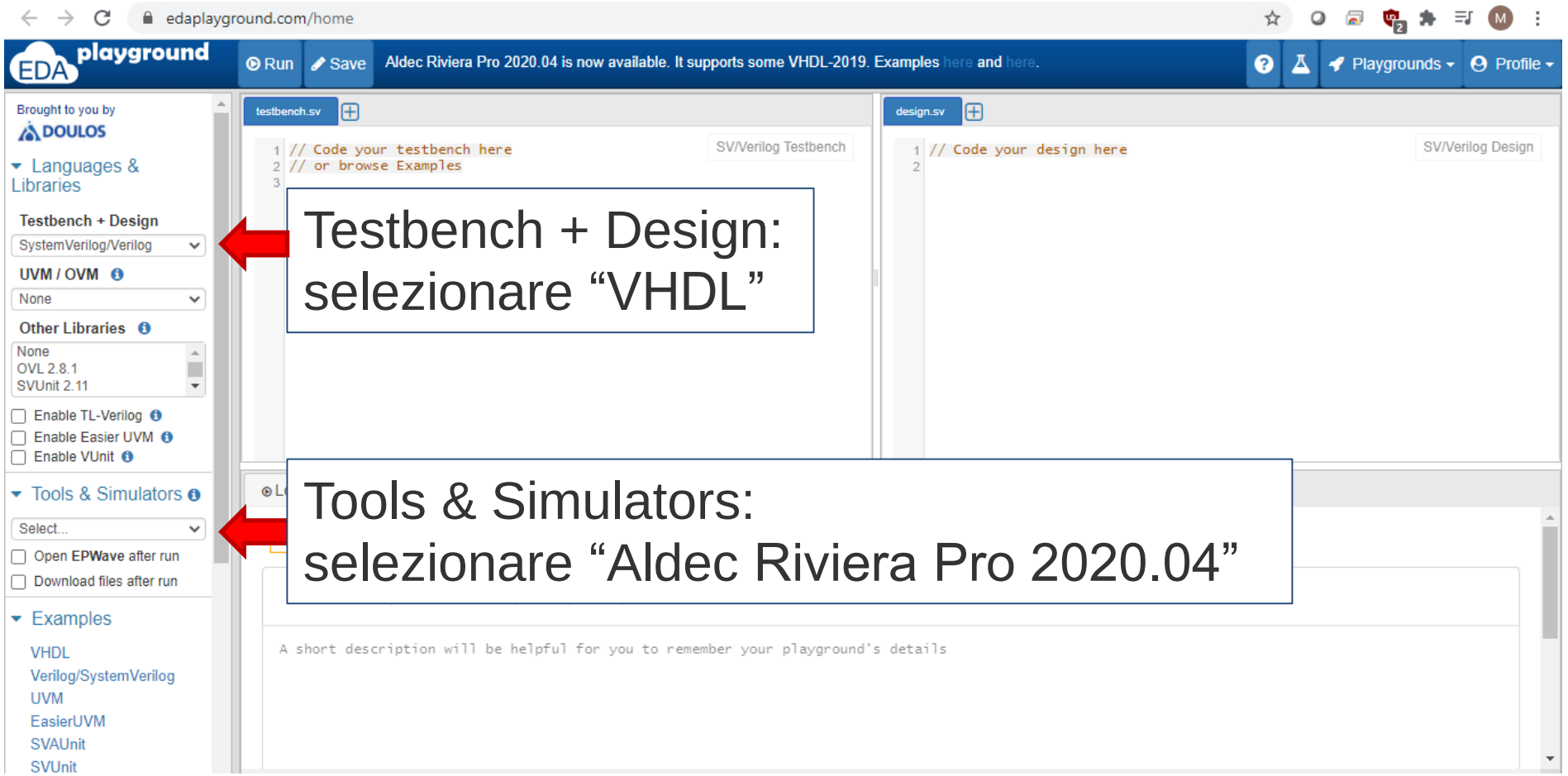
# Scopo della lezione

- Descrivere semplici circuiti logici con **codice VHDL**
- Introdurre il **simulatore VHDL EDA Playground** e simulare il funzionamento di semplici strutture logiche

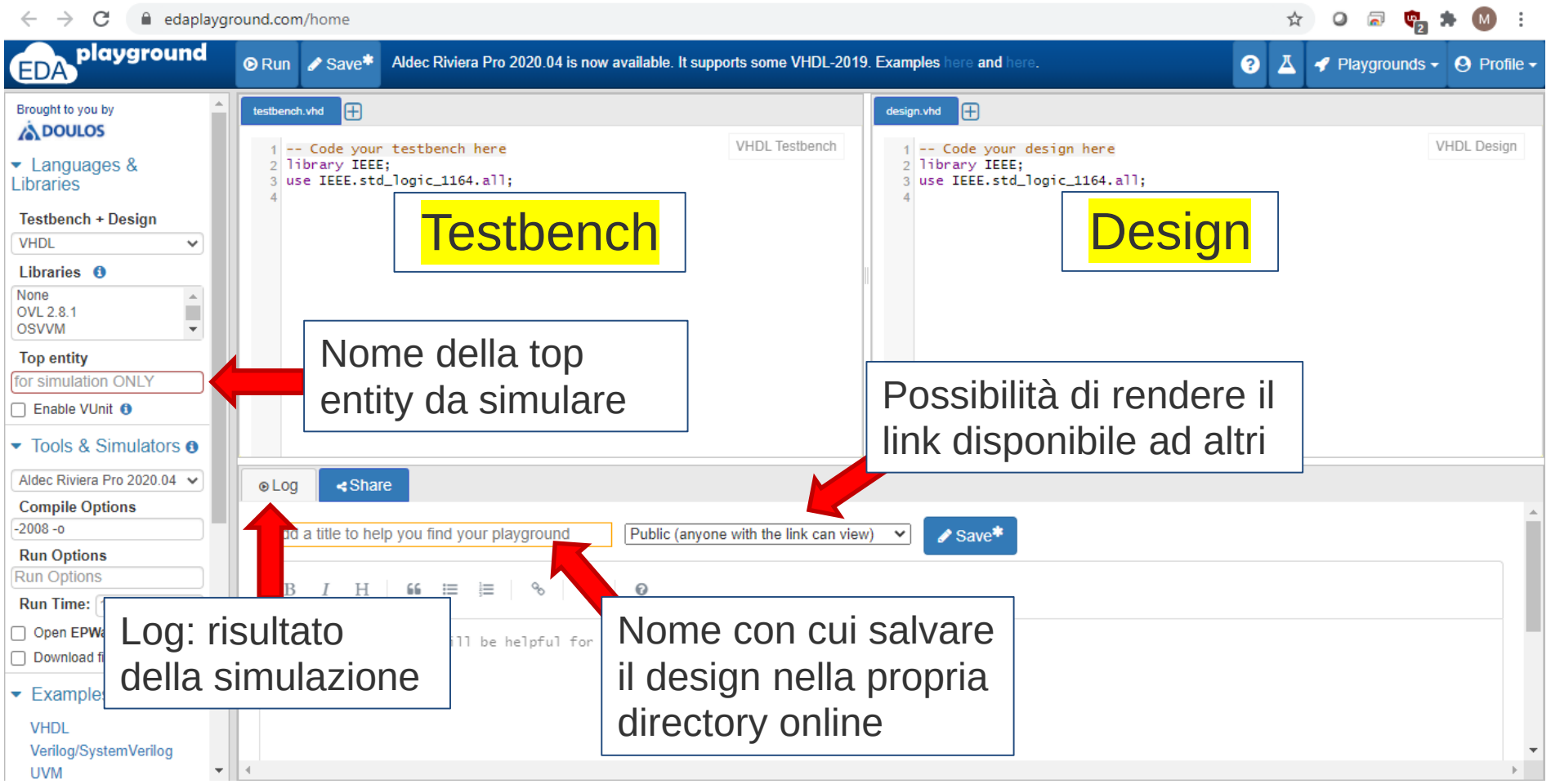
# EDA Playground

- Simulatore online
  - https://www.edaplayground.com/
  - EDA = Electronic Design Automation
  - Non serve installare alcun programma, serve solo un account (anche gmail)
  - Si può lavorare da qualunque PC, una volta che si ha l'account
  - Non serve un computer con grande potenza di calcolo

# EDA Playground

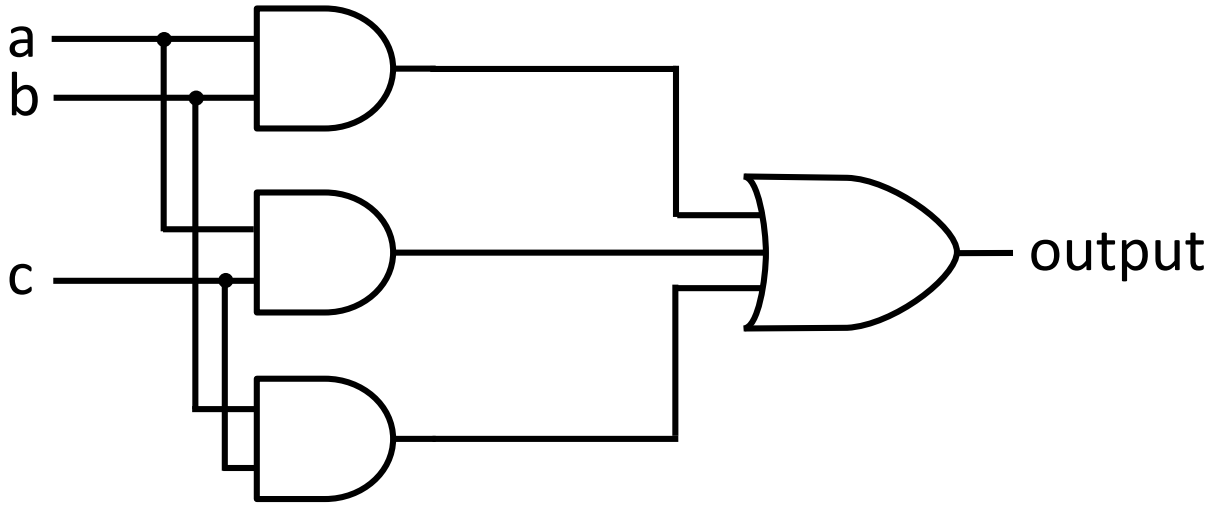


# EDA Playground



# Es: Funzione Majority

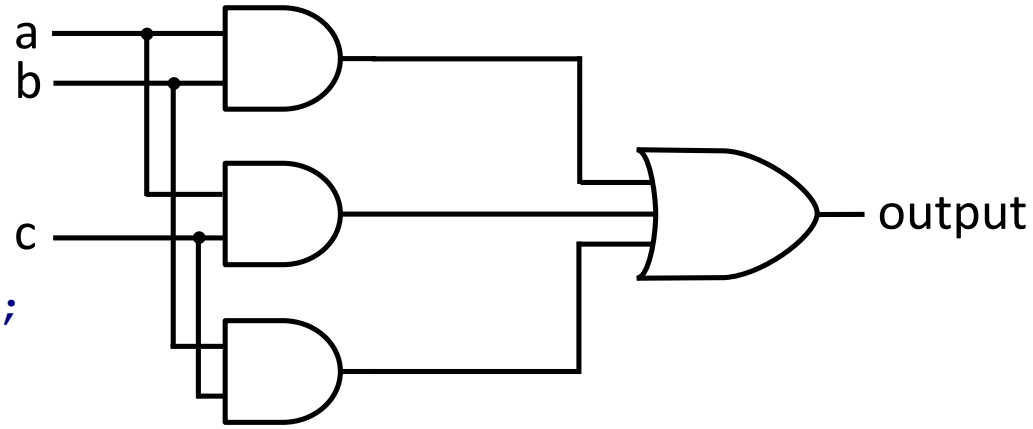
1) Descrivere in VHDL il circuito che descrive la funzione maggioranza
   - 3 ingressi (1 bit) a, b, c
   - 1 uscita (1 bit): output = ‘1’ almeno due degli ingressi valgono ‘1’




2) Simulare il funzionamento del circuito con EDA Playground con una combinazione a piacere degli ingressi

# Funzione Majority: Design

1) Descrivere in VHDL il circuito che descrive la funzione maggioranza



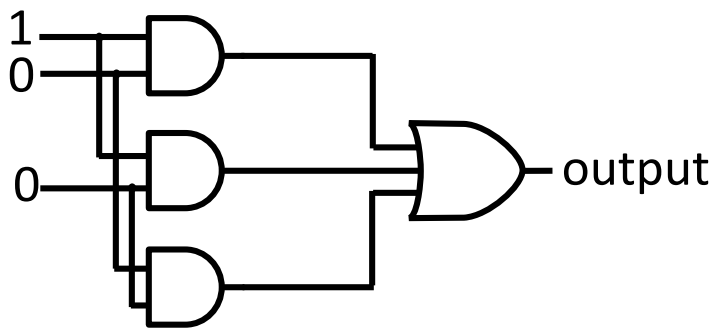

```
library ieee;
use ieee.std_logic_1164.all;

entity majority is
    port ( a, b, c : in std_logic;
           output : out std_logic );
end majority;

architecture maj_dataflow of majority is
    begin
    output <= (a and b) or (a and c) or (b and c);
end maj_dataflow;
```

and e or hanno la stessa precedenza in VHDL (dobbiamo indicare le parentesi) e sono valutati da sinistra verso destra

# Funzione Majority: Testbench

```
library IEEE;
use IEEE.std_logic_1164.all;

entity maj_test is
end maj_test;

architecture test of maj_test is
signal x, y, z, f: std_logic;

component majority is
        port ( a, b, c : in std_logic;
                  output : out std_logic );
end component;
begin
DUT: majority port map (x, y, z, f );
process begin
        x <= '1';   --applica gli stimoli al DUT
        y <= '0';
        z <= '0';
    wait for 10 ns;     --ritardo permette all'uscita di stabilizzarsi
    report "output = " & to_string(f);     --stampa valore dell'uscita
    wait;           --funzione to_string() converte vettore in stringa
end process;
end test;
```

2) Simulare il funzionamento del circuito con EDA Playground con una combinazione a piacere degli ingressi



# Funzione Majority: EDA Playground

testbench.vhd

VHDL Testbench

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity maj_test is
5  end maj_test;
6
7  architecture test of maj_test is
8  signal x, y, z, f: std_logic;
9  component majority is
10     port ( a, b, c : in std_logic;
11            output : out std_logic );
12 end component;
13 begin
14 DUT: majority port map (x, y ,z , f );
15 process begin
16     x <= '1';
17     y <= '0';
18     z <= '0';
19     wait for 10 ns;
20     report "output = " & to_string(f);
21     wait;
22 end process;
23 end test;
```

design.vhd

VHDL Design

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity majority is
5      port ( a, b, c : in std_logic;
6             output : out std_logic );
7  end majority;
8
9  architecture maj_dataflow of majority is
10     begin
11     output <= (a and b) or (a and c) or (b and c);
12 end maj_dataflow;
```

Log | Share

```
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 5399 kB (elbread=427 elab2=4829 kernel=142 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# EXECUTION:: NOTE    : output = 0
# EXECUTION:: Time: 10 ns,  Iteration: 0,  Instance: /maj_test,  Process: line__15.
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
Done
```

# Esercizio 2.34 Mano-Kime

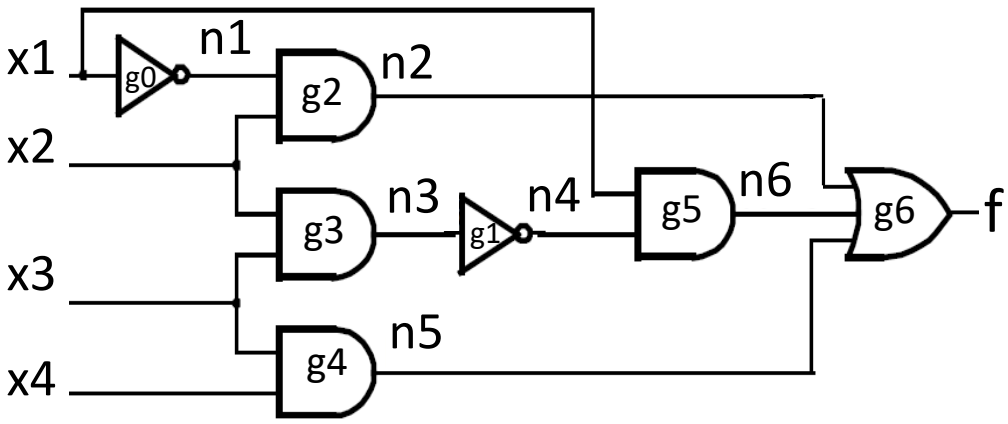
```
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
   entity comb_ckt_1 is
   port(x1, x2, x3, x4 : in std_logic;
        f : out std_logic);
end comb_ckt_1;

architecture structural_1 of comb_ckt_1 is
  component NOT1
    port(in1: in std_logic;
         out1: out std_logic);
  end component;
  component AND2
    port(in1, in2 : in std_logic;
         out1: out std_logic);
  end component;
  component OR3
    port(in1, in2, in3 : in std_logic;
         out1: out std_logic);
  end component;
  signal n1, n2, n3, n4, n5, n6 : std_logic;
  begin
    g0: NOT1 port map (in1 => x1, out1 => n1);
    g1: NOT1 port map (in1 => n3, out1 => n4);
    g2: AND2 port map (in1 => x2, in2 => n1,
                       out1 => n2);
    g3: AND2 port map (in1 => x2, in2 => x3,
                       out1 => n3);
    g4: AND2 port map (in1 => x3, in2 => x4,
                       out1 => n5);
    g5: AND2 port map (in1 => x1, in2 => n4,
                       out1 => n6);
    g6: OR3 port map (in1 => n2, in2 => n5,
                      in3 => n6, out1 => f);
end structural_1;
```

1) Disegnare il diagramma logico del circuito descritto da questo codice VHDL (gli ingressi negati non sono disponibili)

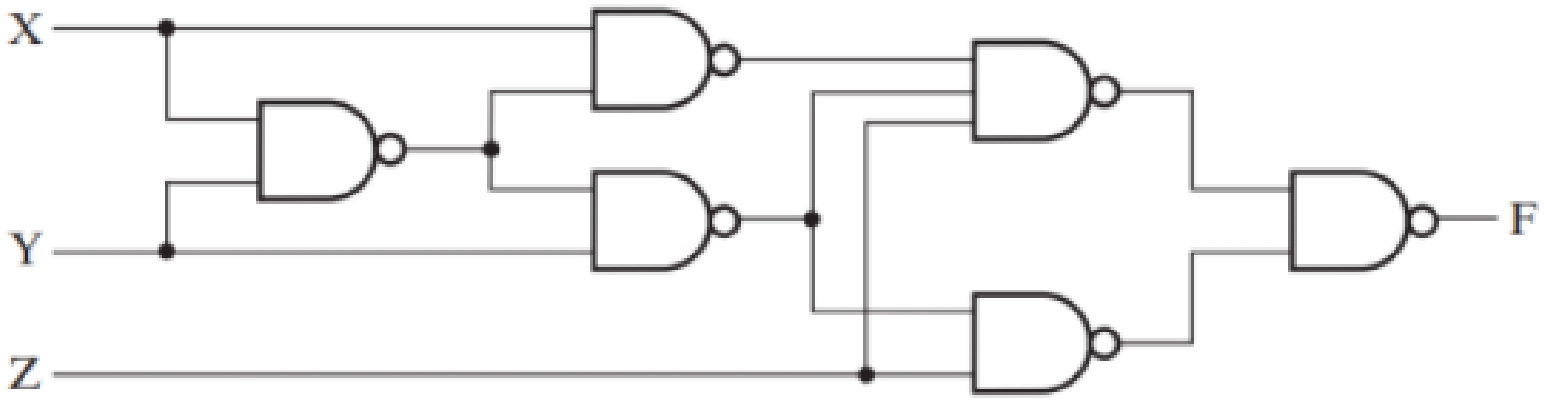
# Esercizio 2.34: diagramma logico

```
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
   entity comb_ckt_1 is
   port(x1, x2, x3, x4 : in std_logic;
        f : out std_logic);
end comb_ckt_1;

architecture structural_1 of comb_ckt_1 is
  component NOT1
    port(in1: in std_logic;
         out1: out std_logic);
  end component;
  component AND2
    port(in1, in2 : in std_logic;
         out1: out std_logic);
  end component;
  component OR3
    port(in1, in2, in3 : in std_logic;
         out1: out std_logic);
  end component;
  signal n1, n2, n3, n4, n5, n6 : std_logic;
  begin
    g0: NOT1 port map (in1 => x1, out1 => n1);
    g1: NOT1 port map (in1 => n3, out1 => n4);
    g2: AND2 port map (in1 => x2, in2 => n1,
                       out1 => n2);
    g3: AND2 port map (in1 => x2, in2 => x3,
                       out1 => n3);
    g4: AND2 port map (in1 => x3, in2 => x4,
                       out1 => n5);
    g5: AND2 port map (in1 => x1, in2 => n4,
                       out1 => n6);
    g6: OR3 port map (in1 => n2, in2 => n5,
                      in3 => n6, out1 => f);
end structural_1;
```

1) Disegnare il diagramma logico del circuito descritto da questo codice VHDL (gli ingressi negati non sono disponibili)

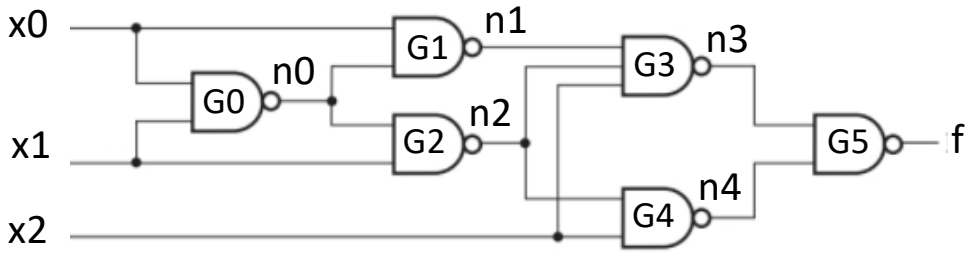


# Esercizio 2.35 Mano-Kime




1) Descrivere il circuito in VHDL con architettura structural e sostituire X, Y, Z con X (2:0)
2) Simulare e verificare il corretto funzionamento del codice con EDA Playground, realizzando un testbench e stimolando il circuito con tutte le possibili combinazioni degli ingressi

# Esercizio 2.35: VHDL design




```
library IEEE;
use IEEE.std_logic_1164.all;

entity circuit2_35 is
   port ( x: in std_logic_vector (2 downto 0);
          f: out std_logic);
end circuit2_35;

architecture circuit2_35_struct of circuit2_35 is
 signal n: std_logic_vector (4 downto 0);
 begin
   G0: entity work.NAND2(NAND2_impl) port map(x(0), x(1), n(0));
   G1: entity work.NAND2(NAND2_impl) port map(x(0), n(0), n(1));
   G2: entity work.NAND2(NAND2_impl) port map(n(0), x(1), n(2));
   G3: entity work.NAND3(NAND3_impl) port map(n(1), n(2), x(2), n(3));
   G4: entity work.NAND2(NAND2_impl) port map(n(2), x(2), n(4));
   G5: entity work.NAND2(NAND2_impl) port map(n(3), n(4), f);

end circuit2_35_struct;
```

# Esercizio 2.35: VHDL testbench

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all; --package numeri senza segno (x addizione)

entity circuit2_35_test is
end circuit2_35_test;

architecture test of circuit2_35_test is
signal input: std_logic_vector (2 downto 0); --per applicare stimoli al DUT
signal output: std_logic;                    --segnale collegato all'uscita dell DUT
component circuit2_35 is
        port ( x: in std_logic_vector (2 downto 0);
               f: out std_logic);
end component;
begin
DUT: circuit2_35 port map(input, output);
process begin    --applica ingressi e stampa valore uscita corrispondente
        input <= "000";
    for i in 0 to 7 loop
        wait for 10 ns;
        report "in = " & to_string(input) & " => out = " & to_string(output);
        input <= input + 1;
    end loop;
    wait;
end process;
end test;
```

# Esercizio 2.35: Log simulazione

Run | Save | Copy | Playgrounds | Profile

testbench.vhd

```
1 -- Testbench Es. 2.35 Mano
2 library IEEE;
3 use IEEE.std_logic_1164.all;
```

VHDL Testbench

design.vhd | MyGates.vhd

```
1 -- Es. 2.35 Mano
2 library IEEE;
3 use IEEE.std_logic_1164.all;
```

VHDL Design

Log | Share

```
# KERNEL: PLI/VHPI kernel's engine initialization done.
# PLI: Loading library '/usr/share/Riviera-PRO/bin/libsystf.so'
# EXECUTION:: NOTE   : in = 000 => out = 0
# EXECUTION:: Time: 10 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 001 => out = 0
# EXECUTION:: Time: 20 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 010 => out = 0
# EXECUTION:: Time: 30 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 011 => out = 0
# EXECUTION:: Time: 40 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 100 => out = 1
# EXECUTION:: Time: 50 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 101 => out = 1
# EXECUTION:: Time: 60 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 110 => out = 0
# EXECUTION:: Time: 70 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 111 => out = 1
# EXECUTION:: Time: 80 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
Finding VCD file...
./dump.vcd
[2021-03-18 09:16:23 EDT] Opening EPWave...
Done
```

# Esercizio 2.35: Forme d'onda

- Selezionando "Open **EPWave** after run" si possono **visualizzare le forme d'onda** dei segnali simulati

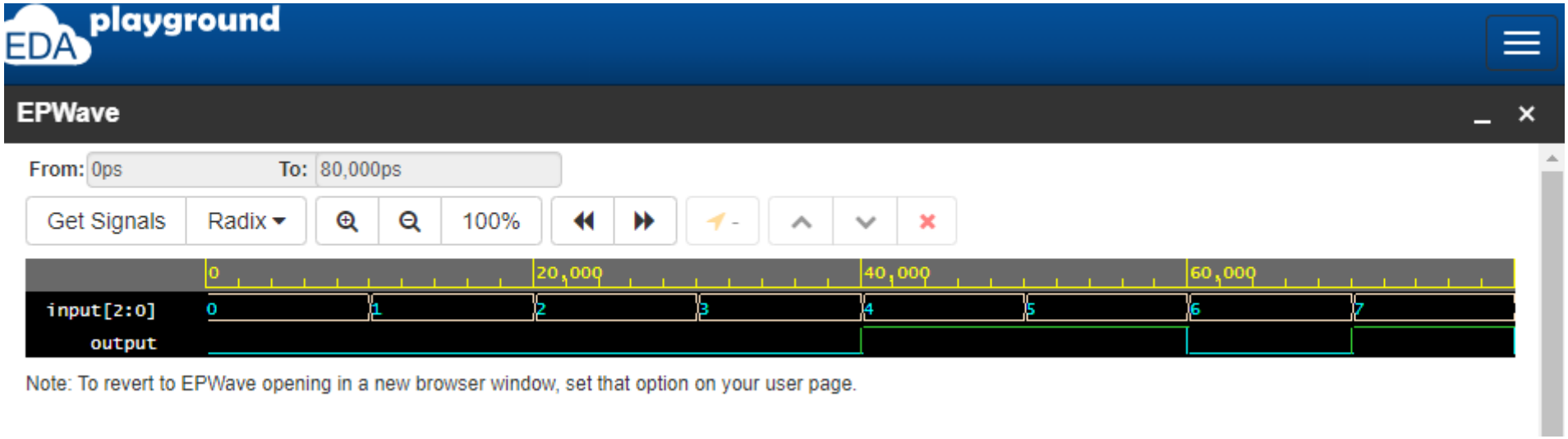



Grafico del valore dell'uscita output in corrispondenza alle 8 possibili combinazioni dei 3 ingressi

# Esercizio 2.37 Mano-Kime

- Trovare il diagramma logico per rappresentare la logica minima a due livelli necessaria ad implementare questa descrizione VHDL dataflow. Gli ingressi sono disponibili in forma diretta e negata.

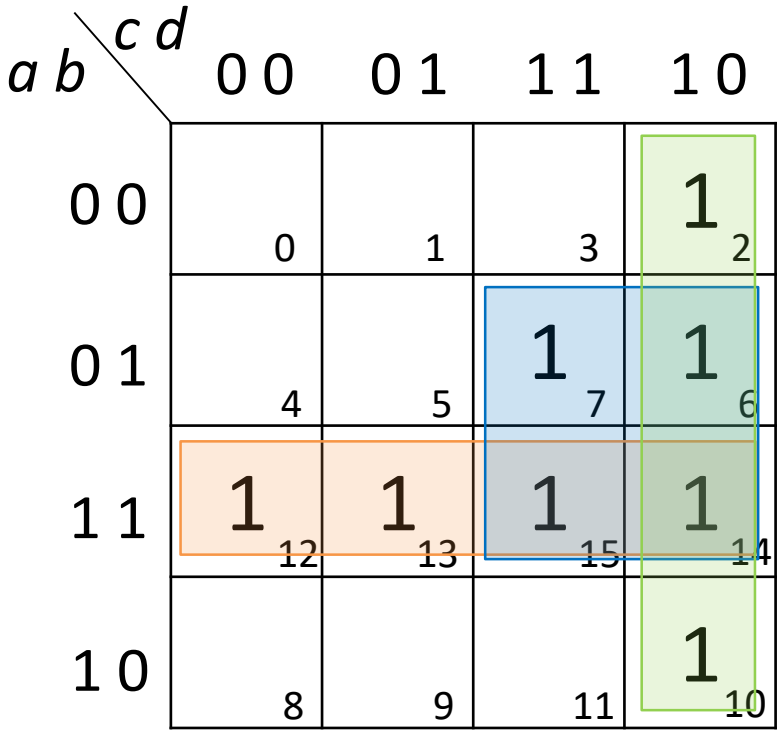
```
-- Combinational Circuit 2: Dataflow VHDL Description
library ieee;
use ieee.std_logic_1164.all;
entity comb_ckt_2 is
   port(a, b, c, d, a_n, b_n, c_n, d_n: in std_logic;
        f, g : out std_logic);
-- a_n, b_n, . . . are complements of a, b, . . . , respectively.

end comb_ckt_2;
architecture dataflow_1 of comb_ckt_2 is
begin
     f <= b and (a or (a_n and c)) or (b_n and c and d_n);
     g <= b and (c or (a_n and c_n) or (c_n and d_n));
end dataflow_1;
```

# Esercizio 2.37

$$f = b \cdot [a + (\bar{a} \cdot c)] + (\bar{b} + c + \bar{d})$$
$$= a \cdot b + \bar{a} \cdot b \cdot c + \bar{b} \cdot c \cdot \bar{d}$$

- Riempiamo la MdK
- Troviamo IP e IPE
- L'espressione minima a due livelli risulta

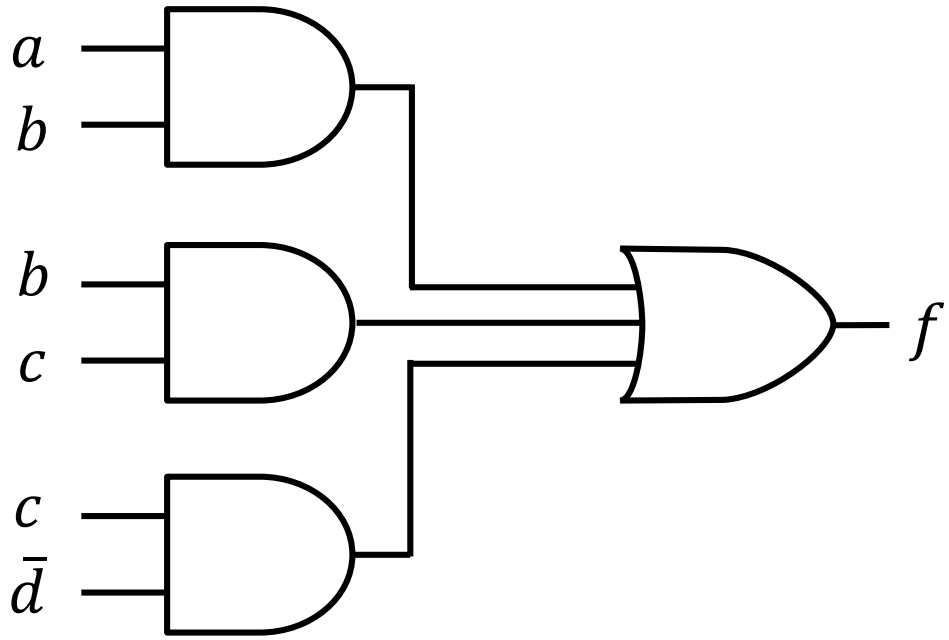
$$f = a \cdot b + b \cdot c + c \cdot \bar{d}$$



# Esercizio 2.37

$$f = a \cdot b + b \cdot c + c \cdot \bar{d}$$

Il diagramma logico per rappresentare la funzione f risulta:



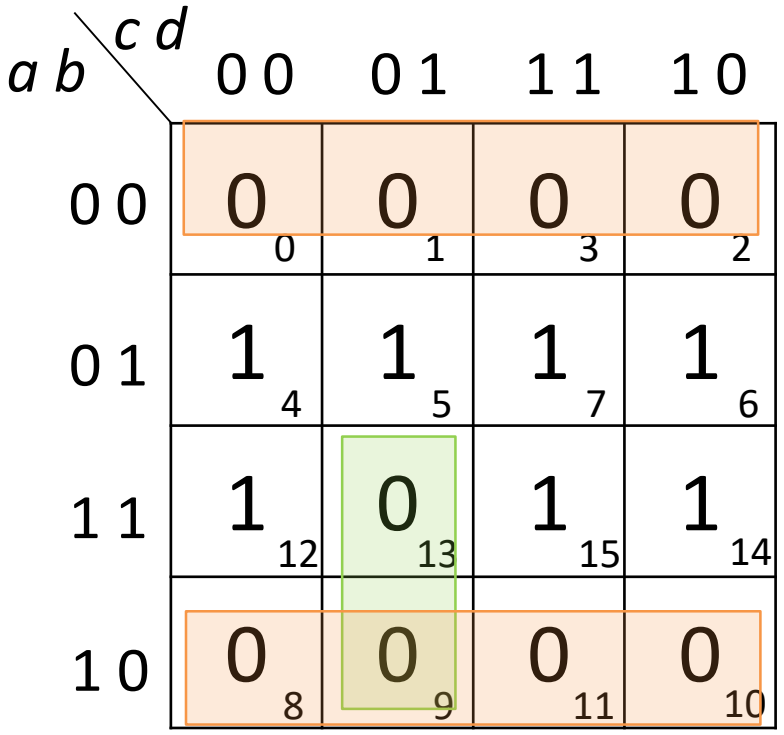
# Esercizio 2.37

$$g = b \cdot [\mathrm{c} + (\bar{a} \cdot \bar{c}) + \bar{c} \cdot \bar{d}]$$
$$= b \cdot c + \bar{a} \cdot b \cdot \bar{c} + b \cdot \bar{c} \cdot \bar{d}$$

- Riempiamo la MdK
- Troviamo IP e IPE: è conveniente lavorare su $\bar{g}$
- L'espressione minima a due livelli risulta

$$\bar{g} = \bar{b} + a \cdot \bar{c} \cdot d$$

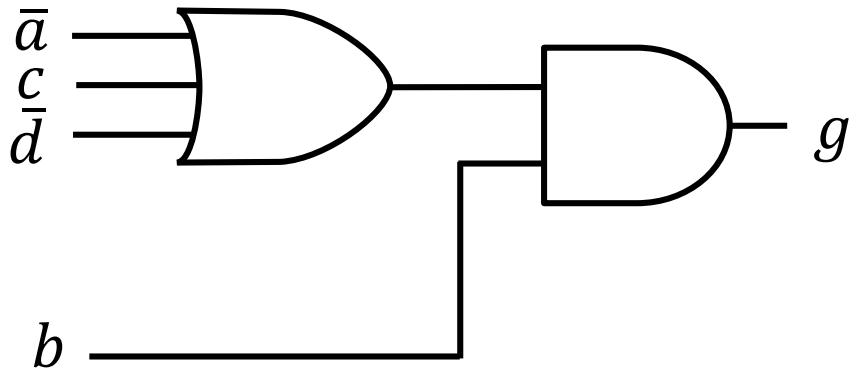
$$\Rightarrow g = b \cdot (\bar{a} + c + \bar{d})$$



# Esercizio 2.37

$$g = b \cdot (\bar{a} + c + \bar{d})$$

Il diagramma logico per rappresentare la funzione g risulta:



# Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime | Martin